

Verification in Staged Tile Self-Assembly

Robert Schweller* Andrew Winslow* Tim Wylie*

Abstract

We prove the unique assembly and unique shape verification problems, benchmark measures of self-assembly model power, are coNP^{NP} -hard and contained in PSPACE (and in Π_{2s}^{P} for staged systems with s stages). En route, we prove that unique shape verification problem in the 2HAM is coNP^{NP} -complete.

1 Introduction

Here we consider the complexity of two standard problems in tile self-assembly: deciding whether a system uniquely assembles a given assembly or shape. These so-called *unique assembly* and *unique shape verification* problems are benchmark problems in tile assembly, and have been studied in a variety of models, including the aTAM [1, 2], the q -tile model [6], and the 2HAM [3].

The unique assembly and unique shape verification problems ask whether a system behaves as expected: does a given system yield a unique given assembly or assemblies of a given unique shape? The distinct rules by which assemblies form in various tile assembly models yield the potential for such problems to have varying complexity. For instance, assuming $\text{P} \neq \text{NP}$, the unique *assembly* verification problem is known to be a strictly easier problem in the aTAM than in the 2HAM.

However, several open questions remain. For instance, such a separation between the aTAM and 2HAM for the unique *shape* verification problem had not been known. Here we prove such a separation (see Table 1).

Additionally, a popular generalization of the 2HAM called the *staged tile assembly model* [7] has been shown to be capable of extremely efficient assembly across a range of parameters [4, 7, 8, 9, 14]. Does this power come from the increased complexity of verifying that systems assemble intended assemblies and shapes?

We achieve progress on these questions, proving a separation between the 2HAM and staged model for the unique assembly verification problem (coNP -complete versus coNP^{NP} -hard) utilizing a promising technique that may lead to proving a stronger separation for

*University of Texas Rio Grande Valley, {robert.schweller, andrew.winslow, timothy.wylie}@utrgv.edu. This research was supported in part by National Science Foundation Grants CCF-1117672 and CCF-1555626.

Model	Unique Assembly	Unique Shape
aTAM	P [1]	coNP-complete [6]
2HAM	coNP-complete [5]	coNP ^{NP} -complete (Sec. 3)
Staged	coNP ^{NP} -hard (Sec. 5), in PSPACE (Sec. 6)	

Table 1: Known and new results on the unique assembly and unique shape verification problems.

the unique shape verification problem (coNP^{NP}-complete versus a conjectured PSPACE-complete).

The coNP^{NP}-hardness results are also interesting as the first, to our knowledge, verification problems in irreversible tile assembly that are decidable but not contained in NP or coNP.

2 The Staged Assembly Model

Tiles. A *tile* is a non-rotatable unit square with each edge labeled with a *glue* from a set Σ . Each pair of glues $g_1, g_2 \in \Sigma$ has a non-negative integer *strength*, denoted $\text{str}(g_1, g_2)$. Every set Σ contains a special *null glue* whose strength with every other glue is 0. If the glue strengths do not obey $\text{str}(g_1, g_2) = 0$ for all $g_1 \neq g_2$, then the glues are *flexible*. Unless otherwise stated, we assume that glues are not flexible.

Configurations, assemblies, and shapes. A *configuration* is a partial function $A : \mathbb{Z}^2 \rightarrow T$ for some set of tiles T , i.e., an arrangement of tiles on a square grid. For a configuration A and vector $\vec{u} = \langle u_x, u_y \rangle \in \mathbb{Z}^2$, $A + \vec{u}$ denotes the configuration $f \circ A$, where $f(x, y) = (x + u_x, y + u_y)$. For two configurations A and B , B is a *translation* of A , written $B \simeq A$, provided that $B = A + \vec{u}$ for some vector \vec{u} . For a configuration A , the *assembly* of A is the set $\tilde{A} = \{B : B \simeq A\}$. An assembly \tilde{A} is a *subassembly* of an assembly \tilde{B} , denoted $\tilde{A} \sqsubseteq \tilde{B}$, provided that there exists an $A \in \tilde{A}$ and $B \in \tilde{B}$ such that $A \subseteq B$. The *shape* of an assembly \tilde{A} is $\{\text{dom}(A) : A \in \tilde{A}\}$ where $\text{dom}()$ is the domain of a configuration. A shape S' is a *scaled* version of shape S provided that for some $k \in \mathbb{N}$ and $D \in S$, $\bigcup_{(x,y) \in D} \bigcup_{(i,j) \in \{0,1,\dots,k-1\}^2} (kx + i, ky + j) \in S'$.

Bond graphs and stability. For a configuration A , define the *bond graph* G_A to be the weighted grid graph in which each element of $\text{dom}(A)$ is a vertex, and the weight of the edge between a pair of tiles is equal to the strength of the coincident glue pair. A configuration is τ -*stable* for $\tau \in \mathbb{N}$ if every edge cut of G_A has strength at least τ , and is τ -*unstable* otherwise. Similarly, an assembly is τ -*stable* provided the configurations it contains are τ -stable. Assemblies \tilde{A} and \tilde{B} are τ -*combinable* into an assembly \tilde{C} provided there exist $A \in \tilde{A}$, $B \in \tilde{B}$, and $C \in \tilde{C}$ such that $A \cup B = C$, $\text{dom}(A) \cap \text{dom}(B) = \emptyset$, and \tilde{C} is τ -stable.

Two-handed assembly and bins. We define the assembly process via bins. A bin is an ordered tuple (S, τ) where S is a set of *initial* assemblies and $\tau \in \mathbb{N}$ is the *temperature*.

In this work, τ is always equal to 2 for upper bounds, and at most some constant for lower bounds. For a bin (S, τ) , the set of *produced* assemblies $P'_{(S, \tau)}$ is defined recursively as follows:

1. $S \subseteq P'_{(S, \tau)}$.
2. If $A, B \in P'_{(S, \tau)}$ are τ -combinable into C , then $C \in P'_{(S, \tau)}$.

A produced assembly is *terminal* provided it is not τ -combinable with any other producible assembly, and the set of all terminal assemblies of a bin (S, τ) is denoted $P_{(S, \tau)}$. That is, $P'_{(S, \tau)}$ represents the set of all possible assemblies that can assemble from the initial set S , whereas $P_{(S, \tau)}$ represents only the set of assemblies that cannot grow any further.

The assemblies in $P_{(S, \tau)}$ are *uniquely produced* iff for each $x \in P'_{(S, \tau)}$ there exists a corresponding $y \in P_{(S, \tau)}$ such that $x \sqsubseteq y$. Unique production implies that every producible assembly can be repeatedly combined with others to form an assembly in $P_{(S, \tau)}$.

Staged assembly systems. An r -stage b -bin mix graph M is an acyclic r -partite digraph consisting of rb vertices $m_{i,j}$ for $1 \leq i \leq r$ and $1 \leq j \leq b$, and edges of the form $(m_{i,j}, m_{i+1,j'})$ for some i, j, j' . A *staged assembly system* is a 3-tuple $\langle M_{r,b}, \{T_1, T_2, \dots, T_b\}, \tau \rangle$ where $M_{r,b}$ is an r -stage b -bin mix graph, T_i is a set of tile types, and $\tau \in \mathbb{N}$ is the temperature. Given a staged assembly system, for each $1 \leq i \leq r$, $1 \leq j \leq b$, a corresponding bin $(R_{i,j}, \tau)$ is defined as follows:

1. $R_{1,j} = T_j$ (this is a bin in the first stage);
2. For $i \geq 2$, $R_{i,j} = \left(\bigcup_{k: (m_{i-1,k}, m_{i,j}) \in M_{r,b}} P_{(R_{i-1,k}, \tau_{i-1,k})} \right)$.

Thus, bins in stage 1 are tile sets T_j , and each bin in any subsequent stage receives an initial set of assemblies consisting of the terminally produced assemblies from a subset of the bins in the previous stage as dictated by the edges of the mix graph.¹ The *output* of a staged system is the union of the set of terminal assemblies of the bins in the final stage.² The output of a staged system is *uniquely produced* provided each bin in the staged system uniquely produces its terminal assemblies.

3 The 2HAM Unique Shape Verification Problem is coNP^{NP} -complete

This section serves as a warm-up for the format and techniques used in later sections. We begin by proving the 2HAM USV problem is in coNP^{NP} by providing a (non-deterministic)

¹The original staged model [7] only considered $O(1)$ distinct tile types, and thus for simplicity allowed tiles to be added at any stage (since $O(1)$ extra bins could hold the individual tile types to mix at any stage). Because systems here may have super-constant tile complexity, we restrict tiles to only be added at the initial stage.

²This is a slight modification of the original staged model [7] in that there is no requirement of a final stage with a single output bin. This may be a slightly more capable model, and so it is considered here. However, all results in this paper apply to both variants of the model.

algorithm for the problem that can be executed on such a machine. This is followed by a reduction from a SAT-like problem complete for coNP^{NP} ($\forall\exists\text{SAT}$).

2HAM unique shape verification (2HAM USV) problem Given a 2HAM system Γ and shape S , does every terminal assembly of Γ have shape S ?

Theorem 3.1. *The 2HAM USV problem (for $\tau = 2$ systems) is coNP^{NP} -hard.*

$\forall\exists\text{SAT}$ Given a 3-SAT formula $\phi(x_1, x_2, \dots, x_k, x_{k+1}, \dots, x_n)$, is it true that for every assignment of x_1, x_2, \dots, x_k , there exists an assignment of $x_{k+1}, x_{k+2}, \dots, x_n$ such that $\phi(x_1, x_2, \dots, x_n)$ evaluates to T?

The $\forall\exists\text{SAT}$ problem was shown to be coNP^{NP} -complete by Stockmeyer [13] (see [12] for further discussion).

Proof. The reduction is from $\forall\exists\text{SAT}$. Roughly speaking, the system output by the reduction behaves as follows. First, a distinct assembly encoding each possible assignment of the variables of the $\forall\exists\text{SAT}$ instance is assembled. Further growth “tags” each assembly as either a *true* or *false* assembly, based upon the truth value of the input 3-SAT formula ϕ for the variable assignment encoded by the assembly.

False assemblies further grow into a slightly larger target shape S . A separate set of *test* assemblies are created, one for each variable assignment of the variables x_1, \dots, x_k . Each test assembly attaches to any true assembly with the same assignment of these variables to form an assembly with shape S - the same shape as false assemblies.

Terminal assemblies then consist of false assemblies and true-test assemblies with shape S , and possibly test assemblies. A test assembly is terminal if and only if there is no true assembly for it to attach to, i.e. the assignment of variables x_1, \dots, x_k has no corresponding assignment of the variables x_{k+1}, \dots, x_n such that $\phi(x_1, \dots, x_n) = \text{T}$.

SAT assemblies. Consider a given input formula C and input value k for the $\forall\exists\text{SAT}$ problem. From this input we design a corresponding 2HAM system $\Gamma = (T, 2)$ and shape S such that the terminal assemblies of Γ share a common shape S if and only if the $\forall\exists\text{SAT}$ instance is “true”, i.e. each assignment of the variables x_1 through x_k can be combined with some assignment of the variables x_{k+1} through x_n such that the 3-SAT instance is satisfied.

The system has temperature 2, and the tile set T of the system output by the reduction is sketched in Figure 1. The first subset of tiles is a minor modification of the commonly used 3-SAT solving system from [11].

For each variable x_i , the system has two tile subsets. These collections assemble into 1×4 assemblies with exposed north and south glues representing the values “0” and “1”, respectively, encoding the assignment of a specific variable to true or false. These 1×4 assemblies further assemble into $1 \times 4n$ assemblies encoding complete assignments of the variables x_1 to x_n . The non-deterministic assembly process of 2HAM implies that such an assembly for every possible variable assignment will be assembled.

An additional column is attached to this bar of height equal to m , the number of clauses in the formula C (Figure 1). An additional set of tiles are added that evaluate the 3-SAT

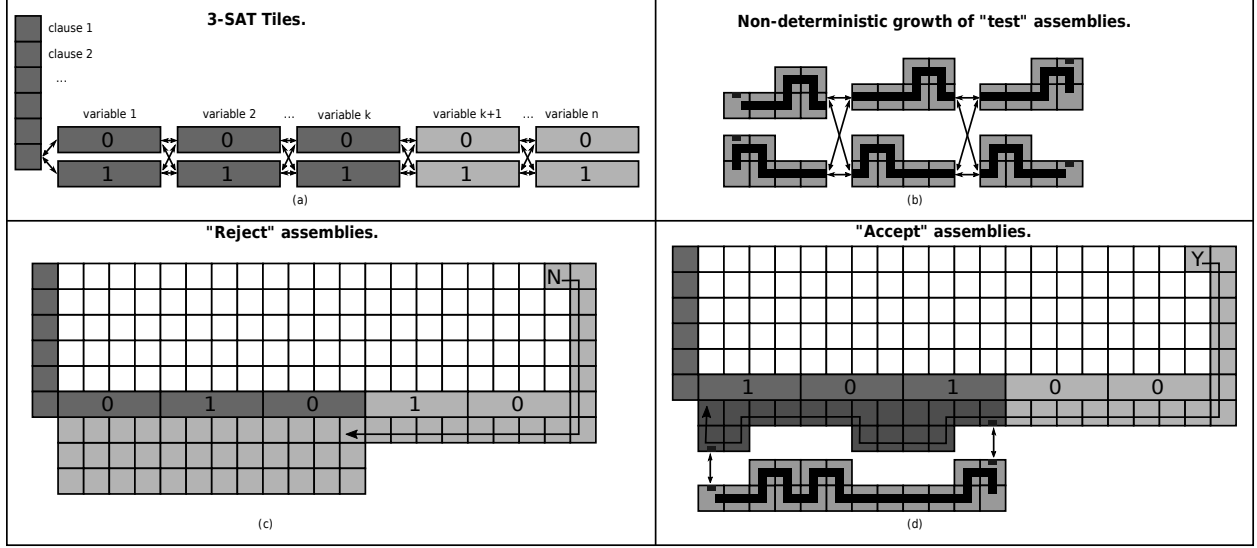


Figure 1: Steps of the 2HAM USV coNP^{NP} -hardness reduction.

formula ϕ based upon the variable assignments encoded by the initial $1 \times 4n$ assembly following the approach of [11]. These tiles place a tile in the upper right corner of the resulting assembly with exposed glue labeled “T” or “F”, indicating the truth value of ϕ based upon the variable assignments.

The resulting assemblies are categorized as *true* and *false* assemblies. Additional tiles are added so that every false assembly further grows, extending the left $4k$ columns (corresponding to the variables x_1 to x_k) southward by 3 rows, and the remaining right $4(n - k)$ columns southward by 1 row (Figure 1(c)). The resulting shape is the shape S output by the reduction, i.e. the only shape assembled by the system if the solution to the $\forall\text{SAT}$ instance is “true”.

Test assemblies. Additional tiles are also added so that true assemblies also grow southward, but extending the left $4k$ columns by various amounts based upon each variable assignment. The result is a sequence of geometric “bumps and dents” that encode the truth values of these variables.

A set of *test* assemblies with complementary geometry for each possible assignment of variables x_1 through x_k are assembled (Figure 1(b)). Test assemblies use two strength-1 glues that cooperatively attach to any true assembly with a matching assignment of variables x_1 through x_k (Figure 1(d)). The assembly formed by a test assembly attaching to a true assembly has shape S : the same shape as a false assembly.

Terminal assemblies. If the solution to the $\forall\text{SAT}$ instance is “false”, there is some truth assignment for variables $x_1 \dots x_k$ with no corresponding assignment of the variables $x_{k+1} \dots x_n$ such that $\phi(x_1, \dots, x_n)$ is “true”. Thus, the test assembly with this assignment of variables x_1, \dots, x_k has no compatible true assembly to attach to - and this test assembly is a terminal assembly of Γ with shape not equal to S .

On the other hand, if the solution to the $\forall\text{SAT}$ instance is “true”, every test assembly

attaches to a true assembly and thus every terminal assembly (true-test assemblies and false assemblies) has shape S . \square

Theorem 3.2. *The 2HAM USV problem is in coNP^{NP} .*

Proof. The solution to an instance (Γ, S) of the 2HAM USV problem is “true” if and only if:

1. Every producible assembly of Γ has size at most $|S|$.
2. Every assembly of size at most $|S|$ and without shape S is not a terminal assembly.

Algorithm 1 solves the 2HAM USV problem by verifying each of these conditions, using an NP subroutine to verify the second condition. The algorithm is executed by a coNP machine, implying that “false” is returned if any of the non-deterministic branches return “false”, and otherwise returns “true”.

Algorithm 1 A coNP^{NP} algorithm for the 2HAM USV problem

- 1: Non-deterministically select a τ -stable assembly A with $|S| < |A| \leq 2|S|$.
 - 2: **if** A is producible **then** \triangleright In P by Theorem 3.2 of [10]
 - 3: **return** false.
 - 4: **end if**
 - 5: Non-deterministically select a τ -stable assembly B with $|B| \leq |S|$ and shape not equal to S .
 - 6: **if** not $\mathcal{F}(\Gamma, B, |S|)$ **then** \triangleright Algorithm 2
 - 7: **return** false.
 - 8: **end if**
 - 9: **return** true.
-

Algorithm 2 An NP algorithm subroutine of Algorithm 1

- 1: **procedure** $\mathcal{F}(\Gamma, B, n)$ \triangleright Returns whether B is *not* terminal.
 - 2: Non-deterministically select a τ -stable assembly C with $|C| \leq n$.
 - 3: **if** C cannot attach to B at temperature τ **then**
 - 4: **return** false.
 - 5: **end if**
 - 6: **if** C is a producible assembly of Γ **then** \triangleright In P by Theorem 3.2 of [10]
 - 7: **return** false.
 - 8: **end if**
 - 9: **return** true.
 - 10: **end procedure**
-

\square

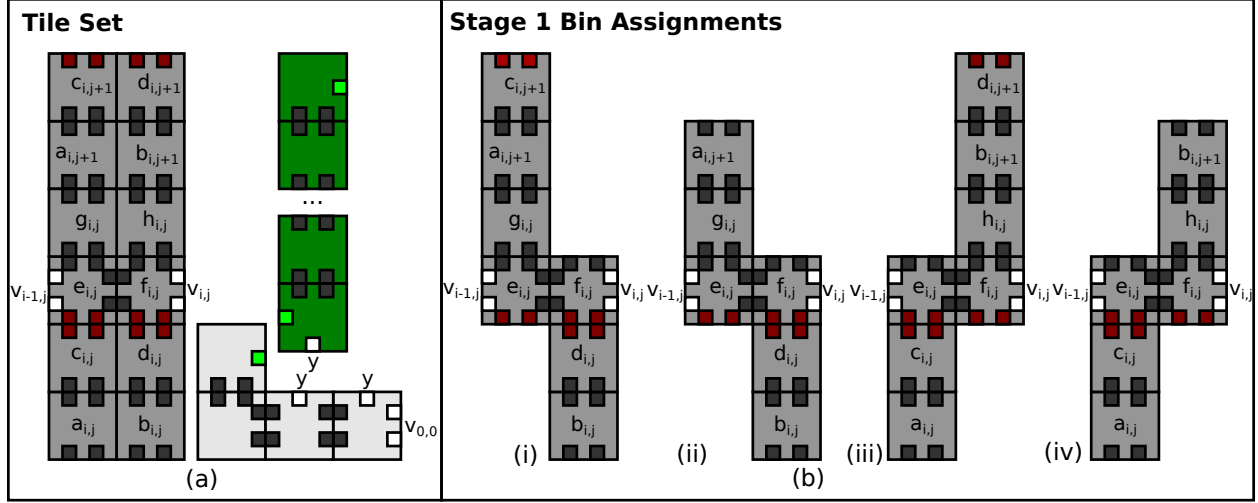


Figure 2: (a) The tile set used in the staged coNP-hardness reduction. (b) The subsets of tiles included in separated initial bins within the first stage of the system.

4 Staged Unique Assembly Verification is coNP-hard

Staged unique assembly verification (Staged UAV) problem Given a staged system Γ and an assembly A , does Γ uniquely assemble A ?

Theorem 4.1. *The staged UAV problem (for 4-stage systems at $\tau = 2$) is coNP-hard.*

Proof. The reduction is from 3-SAT, outputting a staged system Γ and assembly A such that the 3-SAT instance is satisfiable if and only if A is *not* the unique terminal assembly of Γ . We reduce from 3-SAT: Given a 3-SAT formula ϕ , we design a staged assembly system and an assembly A such that ϕ is *not* satisfied if and only if A is uniquely assembled by Γ .

The tileset. The tiles used in our construction are shown in Figure 2(a). In particular, for each variable $x_i \in \{x_1, x_2, \dots, x_n\}$ and clause $c_j \in \{c_1, c_2, \dots, c_m\}$ in ϕ , there is a block of tiles labeled $a_{i,j}, b_{i,j}, c_{i,j}, d_{i,j}, e_{i,j}, f_{i,j}, g_{i,j}$. The set of tile types for each block is denoted $\text{block}_{i,j}$.

The strength-2 ($\tau = 2$) glues connecting adjacent tiles are unique with respect to adjacent tiles, and are unlabelled in the figures for clarity. Note that for each block (i, j) , the top four tiles of the block occupy the same locations as the bottom four tiles of block $(i, j + 1)$. Finally, the tileset includes a length $4m$ chain of *green* tiles, with each green tile sharing a strength-2 glue with its neighbors, along with four light-grey tiles which together attach to the green assembly.

Stage 1: variable assignments. The specific formula ϕ is encoded within the output staged system via the initial choice of tiles placed into a $O(1)$ -sized collection of stage-1 bins. For each variable x_i and clause c_j combination, we select two subsets of the $\text{block}_{i,j}$ tileset. The first subset encodes a variable choice of “false” for x_i . The tile sets in Figure 2(b)(i) and (iv) are used if x_i satisfies (and \bar{x}_i does not satisfy) clause c_j , respectively. Similarly, the tile sets in Figure 2(b)(ii-iii) are used if x_i does not (and \bar{x}_i does satisfy) clause c_j .

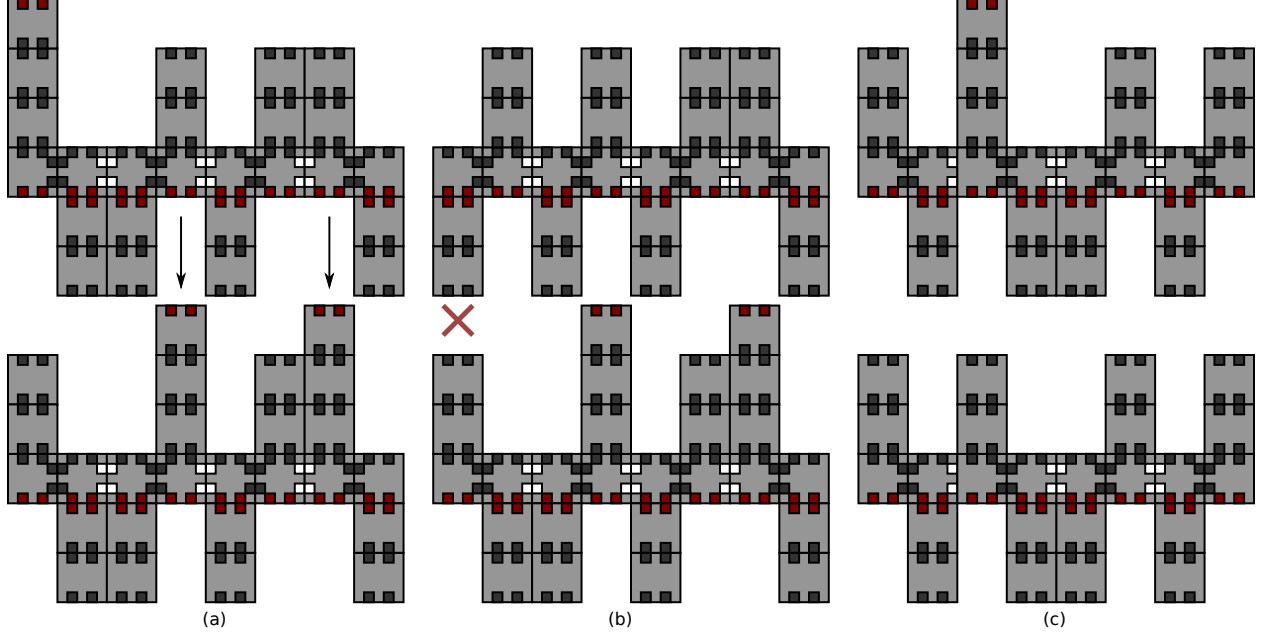


Figure 3: In stage 2, rows non-deterministically form encoding each of the 2^n possible variable assignments. In stage 3 the rows are combined allowing for geometrically compatible, sequential rows with exposed red glue to attach. (a) Combinable rows. (b) Geometrically incompatible rows. (c) Rows with no glues for attachment.

Beyond utilizing two types of $\text{block}_{i,j}$ tile sets, tile sets are further distinguished between odd and even values of i and j . In total, 16 distinct bins (satisfied or not, negated or not, odd or even i , odd or even j) are used.

We include the grey and green tiles of Figure 2(a) separately in two additional bins. An additional four bins are used in the construction to maintain a set of single copies of all tiles used within the system. Separating these tile subsets into four bins ensures that the tiles do no interact (until mixed with other assemblies at a later stage).

Stage 2: assembling rows. In stage 2 we combine all $\text{block}_{i,j}$ assemblies for even j into one bin, and all $\text{block}_{i,j}$ assemblies for odd j into a second bin. Within each bin and for each value j , rows encoding each possible variable assignment assemble non-deterministically via attaching 0 – $\text{block}_{i,j}$ and 1 – $\text{block}_{i,j}$ assemblies for each $i \in \{1, 2, \dots, n\}$. We refer to these assemblies as row_j assemblies. There are 2^n such assemblies for each j - one per variable assignment. Example row_j assemblies are shown in Figure 3.

Stage 3: combining rows with shared assignments and satisfied clauses. Stage 3 is where the *real* action happens. All row_j assemblies are combined, along with the green and grey assemblies of Figure 2.

Consider the possible assembly of a row_j and a row_{j+1} assembly. If the two respective rows encode distinct variable assignments, geometric incompatibility prohibits any possible connection (Figure 3(b)). If the rows encode the same truth assignment, then the rows may attach if any of the row_j variable pieces expose the extended tip via the red $\tau = 2$ strength

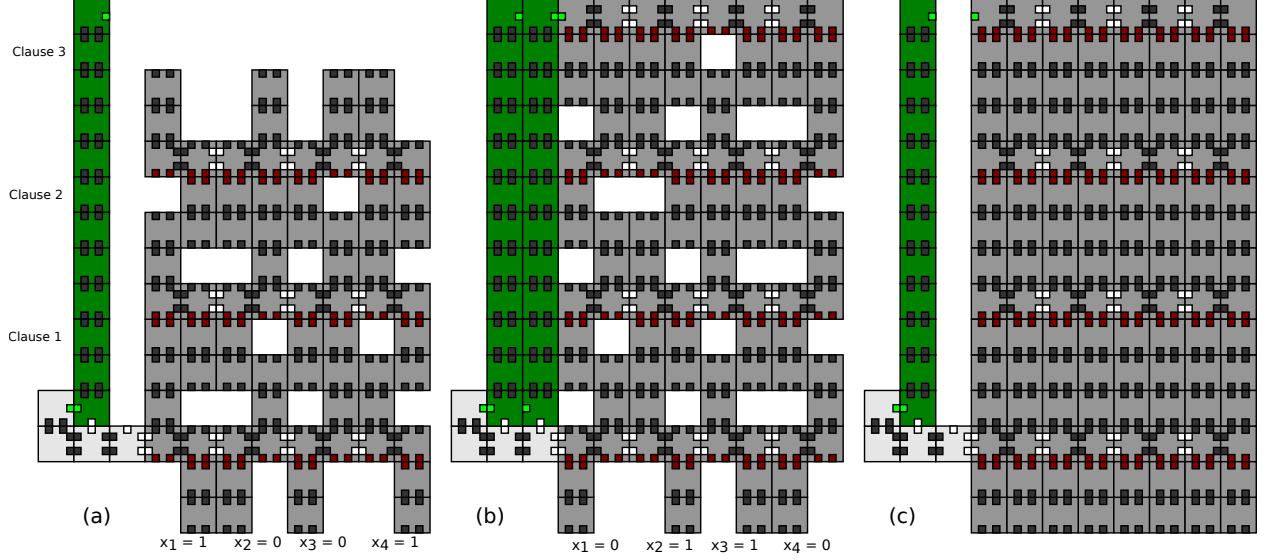


Figure 4: (a) Non-satisfying variable assignments will not be able to grow from row 0 to row m . (b) Assemblies encoding satisfying variable assignments will allow for complete assemblies with all rows, allowing for a green assembly to attach. (c) The target assembly A given as output of the reduction.

glues (Figure 3(a)). Such an attachment indicates that the variable assignment of both rows satisfies c_j . If the variable assignment does not satisfy c_j , no extended tip exists and the rows cannot attach (Figure 3(c)).

A satisfying assignment of ϕ corresponds to m rows attaching to form a complete “satisfying” assembly (Figure 4(b)). The green assembly attaches cooperatively to such assemblies using the row $_m$ assembly glue and a glue from the grey tiles, which attach uniquely to row $_0$. The attachment of a green assembly verifies that all rows are present and the variable assignment satisfies ϕ .

A second copy of the green assembly attaches to any assembly containing row $_0$, regardless of whether all rows are present or not (Figure 4(a)). In a separate bin, the green assembly tiles and grey assemblies are combined, yielding a combined grey-green product (for mixing in stage 4).

Stage 4: merging assignments. In stage 4, the set of all $\text{block}_{i,j}$ individual tiles are added to the assemblies constructed in stage 3 as well as the the grey-green assembly produced in the previous stage. Note that the green assembly is *not* an input assembly to this mixing.

Since all $\text{block}_{i,j}$ assemblies are included, each terminal assembly from stage 3 may grow into the unique terminal assembly shown in Figure 4(c) with one exception: assemblies from stage 3 encoding satisfying variable assignments. These assemblies have one additional copy of the green bar assembly attached. Therefore, the assembly of Figure 4(c) is uniquely assembled if and only if no such satisfying assembly exists. \square

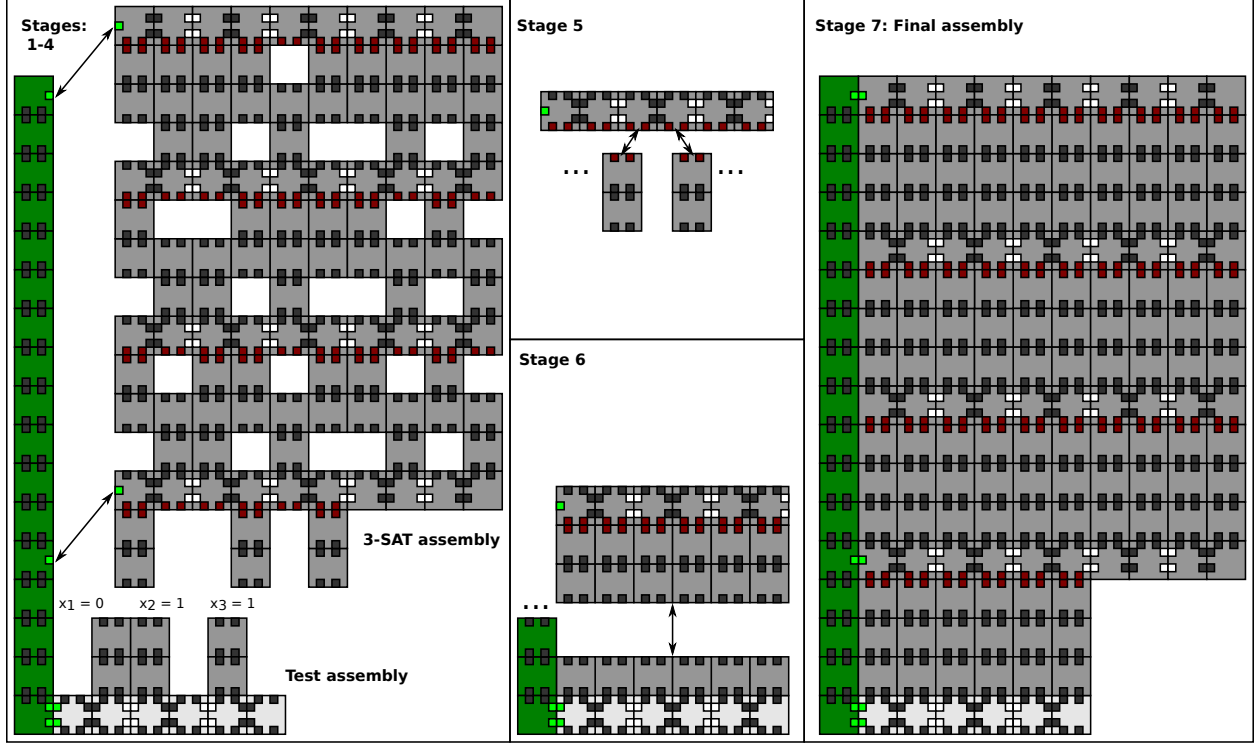


Figure 5: The assemblies at respective stages for the coNP^{NP} -hardness reduction for the staged UAV problem.

5 Staged Unique Assembly Verification is coNP^{NP} -hard

Theorem 5.1. *The staged UAV problem (for $\tau = 2$ 7-stage systems) is coNP^{NP} -hard.*

Proof. We reduce from $\forall\text{SAT}$ by combining ideas from the reductions of Theorem 3.1 and 4.1.

Stages 1-3: the SAT assemblies. The first 3 stages follows those of the reduction in Theorem 4.1 but without the inclusion of the green assembly and light grey tiles. The result is a collection of assemblies encoding satisfying variable assignments with all m rows, as well as partial assemblies of less than m rows encoding non-satisfying assignments. For clarity, the bottom half of the $j = 0$ blocks for values $i > k$ are removed, exposing the “geometric teeth” only for the first k variables.

Stages 1-3: the test assemblies. Additionally, in a separate set of bins, we non-deterministically generate a set of *test* assemblies. The test assemblies are similar to row assemblies and generated in a similar fashion. An example test assembly is shown in Figure 5 (Stages 1-4). A test assembly for each of the 2^k possible truth assignments of x_1, x_2, \dots, x_k is grown, and a green bar assembly is attached to the side of each test assembly.

Stage 4: the magic happens. The SAT assemblies and test assemblies are combined in a bin. Test assemblies attach to SAT assemblies encoding satisfying variable assignments

by utilizing cooperative bonding based on the two strength-1 green glues on the green assembly. SAT assemblies encoding non-satisfying assignments must each lack the topmost or bottommost row, and therefore cannot attach to a test assembly.

Due to the geometric interlocking teeth from the test assembly and the bottom of SAT assemblies, test assemblies may only attach to SAT assemblies that encode the same variable assignment (of variables x_1, x_2, \dots, x_k). Stages 1-4 of Figure 5 show an example test assembly and a attaching SAT assembly.

Note that if there exists a truth assignment for x_1, x_2, \dots, x_k with no satisfying assignment of the remaining variables $x_{k+1}, x_{k+2}, \dots, x_n$, then the corresponding test assembly does not attach to *any* SAT assembly and is a terminal assembly of this bin. On the other had, if every assignment of the variables x_1, x_2, \dots, x_k has at least one satisfying assignment of the remaining variables, i.e. the solution $\forall\exists$ SAT instance is “true”, then there are no terminal test assemblies of this bin

Stage 5: tagging non-satisfying assignments. In Stage 5, we add preassembled duples which attach to the bottom of any assembly containing row 0 and encodes a non-satisfying variable assignment. This attachment ensures that in subsequent stages, these assemblies will be geometrically incompatible with any remaining test assemblies from Stage 4.

It is possible that some duples have no non-satisfying SAT assembly to attach to. As a solution, an additional height-1 assembly of the row-0 assembly that “absorbs” each duple is added at this stage. The subsequent stages enable these, as well as all other SAT assemblies, to grow into a single common (potentially) unique assembly.

Stage 6: attaching test assemblies. The result of Stage 5 is mixed with an assembly consisting of:

- The light-grey bar of the test assemblies.
- A second complete layer of dark grey tiles.
- The green bar.

This assembly attaches to any non-satisfying SAT assembly that includes row 0, ensuring that all assemblies containing row 0 now have a version of the test assembly attached (Stage 6 in Figure 5).

Stage 7: merging. In the final stage, every individual tile of the target assembly (seen in Stage 7 of Figure 5) is added to the result of Stage 6, with the exception of the green tiles and the tiles in rows 1 through 5 of the SAT assemblies.

These tiles complete each SAT assembly in the assembly in Figure 5 (Stage 7). Moreover, the height-1 assembly used to absorb duples from Stage 5 grows into the assembly from Figure 5 (Stage 7). However, because of the lack of tiles from rows 1 through 5, any leftover test assembly from Stage 4 remains terminal.

Thus the target assembly is the unique terminal assembly of the system if and only if the solution to the $\forall\exists$ SAT instance is “yes”. \square

Observe that every staged system output by the reduction has the property that if it does not have a unique terminal assembly, then it also does not have a unique terminal shape. Thus the same reduction suffices to prove that the staged USV problem is coNP^{NP} -hard.

Corollary 5.2. *The staged USV problem is coNP^{NP} -hard.*

6 Staged PSPACE containment

Here we prove that the staged UAV and USV problems are in PSPACE. Parameterized versions of the results are also obtained; these prove that both problems restricted to systems with any *fixed* number of stages lie in the polynomial hierarchy. Both results are obtained via upper bounds on the complexities of the following three problems:

Stage- s producible-in-bin verification (PIBV $_s$) problem Given a staged system Γ , a bin b in stage s of Γ , an assembly A , and an integer n :

1. is A a producible assembly of b ?
2. and does every producible assembly of every bin in stage $s - 1$ of Γ have size at most n ?

Stage- s undersized-in-bin verification (UIBV $_s$) problem Given a staged system Γ , a bin b in stage s of Γ , and an integer n :

1. and does every producible assembly of b have size at most n ?
2. and does every producible assembly of every bin in stage $s - 1$ of Γ have size at most n ?

Stage- s terminal-in-bin verification (TIBV $_s$) problem Given a staged system Γ , a bin b in stage s of Γ , an assembly A , and an integer n :

1. is A a terminal assembly of b ?
2. and does every producible assembly of b have size at most n ?
3. and does every producible assembly of every bin in stage $s - 1$ of Γ have size at most n ?

The statements and proofs of the following results use terminology related to the polynomial hierarchy. For an introduction to the polynomial hierarchy, see Stockmeyer [13]. As a reminder, $\Sigma_{i+1}^{\text{P}} = \text{NP}^{\Sigma_i^{\text{P}}}$, $\Pi_{i+1}^{\text{P}} = \text{coNP}^{\Sigma_i^{\text{P}}}$, and $\Sigma_0^{\text{P}} = \Pi_0^{\text{P}} = \text{P}$.

Lemma 6.1. *For all $s \in \mathbb{N}$:*

- *The PIBV $_s$ problem is in Σ_{2s-2}^{P} .*

- The $UIBV_s$ and $TIBV_s$ problems are in Π_{2s-1}^P .

Due to space limitations, the proof of this lemma is omitted.

Proof. The proof is by induction on s . We begin by proving that $PIBV_1 \in \Sigma_{2s-2}^P = P$ and $UIBV_1, TIBV_1 \in \Pi_{2s-1}^P = \text{coNP}$ (the base case). Then we provide recursive algorithms of the correct complexity for $PIBV_s, UIBV_s$, and $TIBV_s$, assuming that such algorithms exist for $PIBV_{s-1}, UIBV_{s-1}$, and $TIBV_{s-1}$ (the inductive step).

Algorithms for the $PIBV_1, UIBV_1$, and $TIBV_1$ problems. All three problems contain, as a subproblem, “does every producible assembly of every bin in stage $s - 1$ of Γ have size at most n ?”. The answer to this is trivially yes - so only the complexity of the other subproblems needs consideration.

Theorem 3.2 of Doty [10] states that there exists a polynomial-time algorithm for $PIBV_1$. The $UIBV_1$ problem can be solved by a **coNP** machine via non-deterministically selecting an assembly of size in $(n, 2n]$ consisting of tile types input into bin b and returning “no” if the assembly is producible (the machine returns “no” if any non-deterministic branch returns “no”). The $TIBV_1$ problem can be solved by a **coNP** machine by (1) returning “no” if A is not producible, (2) returning “no” if a second assembly (non-deterministically selected) is producible and attaches to A , (3) returning “yes” otherwise.

An algorithm for the $PIBV_s$ problem. We now assume from now on that there exist algorithms $\mathcal{P}_{s-1}, \mathcal{U}_{s-1}$, and \mathcal{T}_{s-1} for the $PIBV_{s-1}, UIBV_{s-1}$, and $TIBV_{s-1}$ problems in $\Sigma_{2s-4}^P, \Pi_{2s-3}^P$, and Π_{2s-3}^P , respectively, by the inductive hypothesis.

Algorithm 3 A Σ_{2s-2}^P algorithm for the PIBV_s problem

```

1: procedure  $\mathcal{P}_s(\Gamma, b, A, n)$  ▷ Bin  $b$  is in stage  $s$  of  $\Gamma$ 
2:   if not  $A$  is  $\tau$ -stable then ▷ In  $P$  via min-cut
3:     return no.
4:   end if
5:    $I \leftarrow \{A\}$ 
6:   while non-deterministically choosing to continue and  $|I| < |A|$  do
7:     Decompose an assembly  $B$  in  $I$  into two stable subassemblies  $B_1, B_2$ .
8:      $I = (I - B) \cup \{B_1, B_2\}$  ▷ Replace  $B$  with  $B_1$  and  $B_2$ 
9:   end while
10:  Non-deterministically assign a bin  $b_{B_i}$  in stage  $s - 1$  to each  $B_i \in I$ .
11:  for all  $B_i \in I$  do
12:    if not  $\mathcal{T}_{s-1}(\Gamma, b_{B_i}, B_i, n)$  then ▷ Function call is in  $\Pi_{2s-3}^P$ 
13:      return no.
14:    end if
15:  end for
16:  for all bins  $b'$  in stage  $s - 1$  do ▷ Subproblem 2
17:    if not  $\mathcal{U}_{s-1}(\Gamma, b', n)$  then ▷ Function call is in  $\Pi_{2s-3}^P$ 
18:      return no.
19:    end if
20:  end for
21:  return yes.
22: end procedure

```

The algorithm runs as an NP machine (making calls to other machines). Lines 5-10 non-deterministically compute an assembly process for A in bin b , and lines 8-12 check that such a process begins with terminal assemblies of (specific) input bins. Lines 13-18 simply check that the condition of subproblem 2 is satisfied.

The complexity of the algorithm is NP with polynomially many calls to algorithms in Π_{2s-3}^P . That is, $\text{NP}^{\Pi_{2s-3}^P} = \text{NP}^{\Sigma_{2s-3}^P} = \Sigma_{2s-2}^P$.

An algorithm for the UIBV_s problem. Since we have already proved that there exists a Σ_{2s-2}^P algorithm \mathcal{P}_s , we assume this as well.

Algorithm 4 A Π_{2s-1}^P algorithm for the UIBV_s problem

```

1: procedure  $\mathcal{U}_s(\Gamma, b, n)$  ▷ Bin  $b$  is in stage  $s$  of  $\Gamma$ 
2:   Non-deterministically select an assembly  $A$  with  $n < |A| \leq 2n$ .
3:   if  $\mathcal{P}_s(\Gamma, b, A, n)$  then ▷ Function call is in  $\Sigma_{2s-2}^P$ 
4:     return no.
5:   end if
6:   for all bins  $b'$  in stage  $s - 1$  do
7:     if  $\mathcal{P}_s(\Gamma, b', A, n)$  then ▷ Function call is in  $\Sigma_{2s-4}^P$ 
8:       return no.
9:     end if
10:  end for
11:  return yes.
12: end procedure

```

The algorithm runs as a **coNP** machine, returning “no” unless every non-deterministic branch returns “yes”. Lines 2-5 solve subproblem 1, while lines 6-10 address subproblem 2.

The complexity of the algorithm is then **coNP** with two calls to algorithms in Σ_{2s-2}^P . That is, $\text{coNP}^{\Sigma_{2s-2}^P} = \Pi_{2s-1}^P$.

An algorithm for the TIBV_s problem. Since we have already proved that there exists a Π_{2s-1}^P algorithm \mathcal{U}_s , we assume this as well.

Algorithm 5 An Π_{2s-1}^P algorithm for the TIBV_s problem

```

1: procedure  $\mathcal{T}_s(\Gamma, b, A, n)$  ▷ Bin  $b$  is in stage  $s$  of  $\Gamma$ 
2:   if not  $\mathcal{P}_s(\Gamma, b, A, n)$  then ▷ Function call in  $\Sigma_{2s-2}^P$ 
3:     return no.
4:   end if
5:   Non-deterministically select an assembly  $B$  with  $|B| \leq n$ .
6:   if  $\mathcal{P}_s(\Gamma, b, B, n)$  and  $A$  and  $B$  can attach at temperature  $\tau$  then
7:     return no.
8:   end if
9:   if not  $\mathcal{U}_s(\Gamma, b, n)$  then ▷ Subproblems 2 and 3
10:    return no.
11:  end if
12:  return yes.
13: end procedure

```

The algorithm runs as a **coNP** machine, returning “no” unless every non-deterministic branch returns “yes”. Lines 2-8 verify that A is a terminal assembly of bin b (subproblem 1): A is not a terminal assembly if and only if (1) A is not producible (lines 2-4), or (2) another producible assembly B can attach to A (lines 5-8).

The complexity of the algorithm needs a slightly careful analysis. Lines 2-8 can be seen as a **coNP** algorithm with two calls to algorithms in Σ_{2s-2}^P , i.e. a $\text{coNP}^{\Sigma_{2s-2}^P} = \Pi_{2s-1}^P$ algorithm.

Then the entire algorithm is a P algorithm with a call to a Π_{2s-1}^P algorithm (lines 2-8) and another call to a Π_{2s-1}^P algorithm (line 9). That is, a $P^{\Pi_{2s-1}^P} = \Pi_{2s-1}^P$ algorithm.

A remark on the reoccurring subproblem. All three problems have the subproblem “does every producible assembly of every bin in stage $s - 1$ of Γ have size at most n ?” Removing this subproblem from the $TIBV_s$ problem makes the problem undecidable, since arbitrarily large assemblies (carrying out unbounded computation) may attach to A . Seen from another perspective, line 5 of \mathcal{T}_s is only correct because we may assume that any attaching assembly B has size at most n . The $PIBV_s$ and $UIBV_s$ problems are also similarly undecidable when the subproblem is removed.

In a system with a unique terminal assembly/shape, no producible assembly of any bin has size exceeding that the unique terminal assembly/shape. Thus adding such a subproblem does not change the answer to staged UAV/USV problem instances (a “no” with the added subproblem implies a “no” without it as well). \square

With this algorithmic machinery in place, we move to the first main result:

Stage- s unique assembly verification (Stage- s UAV) problem Given a staged system Γ with s stages and an assembly A , is A the unique terminal assembly of Γ ?

Theorem 6.2. *The stage- s UAV problem is in Π_{2s}^P .*

Proof. We give an algorithm for the stage- s UAV problem. The stage- s UAV problem may be restated as:

1. is every assembly B with $|B| \leq |A|$ and $B \neq A$ not a terminal assembly of any bin in stage s ?
2. and does every producible assembly of every bin in stage $s - 1$ of Γ have size at most $|A|$?

In the algorithm below, \mathcal{T}_s and \mathcal{U}_s are algorithms for the $TIBV_s$ and $UIBV_s$ problems, respectively.

Algorithm 6 A Π_{2s}^P algorithm for the stage- s UAV problem

```

1: procedure  $\mathcal{UAV}_s(\Gamma, A)$  ▷  $\Gamma$  has  $s$  stages.
2:   Non-deterministically select an assembly  $B$  with  $|B| \leq n$  and  $A \neq B$ .
3:   for all bins  $b$  in stage  $s$  of  $\Gamma$  do
4:     if  $\mathcal{T}_s(\Gamma, b, B)$  then ▷ Function call is in  $\Pi_{2s-1}^P$ 
5:       return no.
6:     end if
7:   end for
8:   if not  $\mathcal{U}_s(\Gamma, b, |A|)$  then ▷ Function call is in  $\Pi_{2s-1}^P$ 
9:     return no.
10:  end if
11:  return yes.
12: end procedure

```

The algorithm runs as a **coNP** machine, returning “no” unless every non-deterministic branch returns “yes”. Lines 2-8 verify that A is a terminal assembly of bin b (subproblem 1): A is not a terminal assembly if and only if (1) A is not producible (lines 2-4), or (2) another producible assembly B can attach to A (lines 5-8). \square

Every staged system has some number of stages $s \in \mathbb{N}$, but there is no limit to the number of stages a staged system may have. Thus the staged UAV problem is not contained in any level of **PH**, but every instance can be solved by an algorithm that runs at a fixed level (Π_{2s}^P) of the hierarchy. Since it is a well-known that $\text{PH} \subseteq \text{PSPACE}$, this gives the desired result:

Corollary 6.3. *The staged UAV problem is in **PSPACE**.*

Next, we move to shape verification:

Stage- s unique shape verification (Stage- s USV) problem Given a staged system Γ with s stages and a shape S , is S the unique terminal shape of Γ ?

Theorem 6.4. *The stage- s USV problem is in Π_{2s}^P .*

Proof. The stage- s USV problem can be restated as:

1. is every assembly B with $|B| \leq |S|$ and shape not equal to S not a terminal assembly of any bin in stage s ?
2. and does every producible assembly of every bin in stage $s - 1$ of Γ have size at most $|S|$?

Notice that the subproblems only differ from those of the stage- s UAV problem in that S replaces A and “equal shape” replaces “equals”. Thus the algorithm differs from the Π_{2s}^P algorithm for the stage- s UAV problem on only line 5 (replace “ $A \neq B$ ” with “shape not equal to S ”) and line 8 (replace $|A|$ with $|S|$). \square

As for the UAV problem, since the stage- s USV problem is in **PH** for each $s \in \mathbb{N}$, the USV problem is in **PSPACE**.

Corollary 6.5. *The staged USV problem is in **PSPACE**.*

7 Open Problems

The most direct problem left open by this work is closing the gap in the bottom row of Table 1 between the **coNP^{NP}**-hardness and **PSPACE** containment of the staged UAV and USV problems. We believe that the approach of differentiating between satisfying and non-satisfying assignments, then checking for the existence of various partial assignments (the \forall portion of $\forall\exists\text{SAT}$) can be generalized to achieve hardness for any number of quantifier alternations, using a number of stages proportional to the number of alternations:

Conjecture 7.1. *The staged UAV and USV problems are PSPACE-complete.*

Conjecture 7.2. *The stage- s UAV and stage- s USV problems are $\Pi_{\Omega(s)}^p$ -hard.*

The UAV and USV problems considered in this work are two variants of the generic challenge of *verification*; considering the same problems limited to temperature-1 systems or with different inputs is also interesting:

Problem 7.3. *What are the complexities of the staged UAV and USV problems restricted to temperature-1 systems?*

Problem 7.4. *What is the complexity (in any model) of the following UAV-like problem: given a system Γ and an integer n , does Γ have a unique terminal assembly of size at most n ?*

Finally, the results and techniques presented here might find use in the study of other problems in staged and two-handed self-assembly, such as tile minimization. The aTAM USV problem is coNP-complete, while the *minimum tile set problem* of finding the minimum number of tiles that uniquely assemble into a given shape is NP^{NP} -complete [2]. We now know that the 2HAM USV problem is coNP^{NP}-complete (Section 3); does the corresponding optimization problem also rise in the hierarchy?

Conjecture 7.5. *The 2HAM minimum tile set problem is $\text{NP}^{\text{NP}^{\text{NP}}}$ -complete.*

References

- [1] L. M. Adleman, Q. Cheng, A. Goel, M.-D. A. Huang, D. Kempe, P. M. de Espanés, and P. W. K. Rothmund. Combinatorial optimization problems in self-assembly. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing*, pages 23–32, 2002.
- [2] N. Bryans, E. Chiniforooshan, D. Doty, L. Kari, and S. Seki. The power of nondeterminism in self-assembly. *Theory of Computing*, 9(1):1–29, 2013.
- [3] S. Cannon, E. D. Demaine, M. L. Demaine, S. Eisenstat, M. J. Patitz, R. T. Schweller, S. M. Summers, and A. Winslow. Two hands are better than one (up to constant factors): Self-assembly in the 2HAM vs. aTAM. In *STACS 2013*, volume 20 of *LIPIcs*, pages 172–184. Schloss Dagstuhl, 2013.
- [4] C. Chalk, E. Martinez, R. Schweller, L. Vega, A. Winslow, and T. Wylie. Optimal staged self-assembly of general shapes. In *Proc. of the 24th European Symposium of Algorithms*, volume 57 of *LIPIcs*, pages 26:1–26:17. Schloss Dagstuhl, 2016.
- [5] C. Chalk, R. Schweller, A. Winslow, and T. Wylie. Too hot 2HAMdle: high-temperature two-handed self-assembly. Under submission, 2017.

- [6] Q. Cheng, G. Aggarwal, M. H. Goldwasser, M.-Y. Kao, R. T. Schweller, and P. M. de Espanés. Complexities for generalized models of self-assembly. *SIAM Journal on Computing*, 34:1493–1515, 2005.
- [7] E. D. Demaine, M. L. Demaine, S. P. Fekete, M. Ishaque, E. Rafalin, R. T. Schweller, and D. L. Souvaine. Staged self-assembly: nanomanufacture of arbitrary shapes with $O(1)$ glues. *Natural Computing*, 7(3):347–370, 2008.
- [8] E. D. Demaine, S. Eisenstat, M. Ishaque, and A. Winslow. One-dimensional staged self-assembly. In *Proceedings of the 17th international conference on DNA computing and molecular programming*, DNA’11, pages 100–114, 2011.
- [9] E. D. Demaine, S. P. Fekete, C. Scheffer, and A. Schmidt. New geometric algorithms for fully connected staged self-assembly. In *DNA Computing and Molecular Programming*, volume 9211 of *LNCS*, pages 104–116. Springer, 2015.
- [10] D. Doty. Producibility in hierarchical self-assembly. In *Proceedings of Unconventional Computation and Natural Computation (UCNC)*, volume 8553 of *LNCS*, pages 142–154. Springer, 2014.
- [11] M. G. Lagoudakis and T. H. Labean. 2d dna self-assembly for satisfiability. In *5th International Meeting on DNA Based Computers*, 1999.
- [12] M. Schaefer and C. Umans. Completeness in the polynomial-time hierarchy: a compendium. *SIGACT News*, 33(3):32–49, 2002.
- [13] L. J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.
- [14] A. Winslow. Staged self-assembly and polyomino context-free grammars. *Natural Computing*, 14(2):293–302, 2015.